DOCUMENT RESUME

ED 267 748 IR 012 068

AUTHOR	Cohen, Stanley
TITLE	Multiplan as a Programming Language. Microcomputing Working Paper Series.
INSTITUTION	Drexel Univ., Philadelphia, PA. Microcomputing Program.
REPORT NO	MWPS-F-84-11
PUB DATE	Dec 84
NOTE	33p.
AVAILABLE FROM	Working Paper Series, Microcomputing Program, Office of the Vice President for Academic Affairs, Drexel University, Philadelphia, PA 19104.
PUB TYPE	Guides - Non-Classroom Use (055) -- Viewpoints (120) -- Reports - Descriptive (141)
EDRS PRICE	MF01/PC02 Plus Postage.
DESCRIPTORS	*Computer Software; Higher Education; Instructional Materials; *Microcomputers; Program Descriptions; *Programing; *Programing Languages
IDENTIFIERS	*Drexel University PA; *Spreadsheets

ABSTRACT

This description of ways in which a computer spreadsheet program, Multiplan, can be used as a programming language begins by comparing the use of two programming languages--Pascal and Multiplan--to solve a simple problem. Next, the advantages of using a spreadsheet are described; e.g., keyboard input and screen formatting routines are handled by the program; the spreadsheet is completely interactive; the cell format encourages partitioning tasks into small modules; Multiplan allows cells or groups of cells to be given meaningful names; calculations requiring iterative solutions can be performed; and results of calculations of one sheet can be transferred to another sheet. The limitations of using the spreadsheet as a programming language are then described; e.g., as the sheet fills up recalculation takes longer, the fixed format is limiting, and the implementation of iteration on Multiplan forces the order of calculation of cells to occur in a way which restricts the positioning of formulas on the sheet. Finally, several examples of programs done with Multiplan are briefly described. Appendices include examples of nine Multiplan template set-up sheets and screens. (THC)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
OERI
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official position or policy.

Multipian® As A Programming Language

Stanley Cohen
Dept. of Physics & Atmospheric Sciences
Drexel University

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Lucy A. Boardway
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Multiplan® As A Programming Language

Stanley Cohen
Dept. of Physics & Atmospheric Sciences
Drexel University

MWPS F 84-3
December, 1984

Requests for copies of papers in the Microcomputing Working Paper Series should be sent to: Working Paper Series, Microcomputing Program, Office of the Vice President for Academic Affairs, Drexel University, Philadelphia, PA 19104.

Multiplan® As A Programming Language

Introduction

The computer is a machine with an adjustable personality. Each application presents the user with a new "device" or more precisely the illusion of a device. A programmer using a high level programming language such as Pascal, BASIC, FORTRAN or FORTH is deluding himself if he thinks he is programming a computer. These programming languages provide the user with a model of an ideal computer. Programs written in high level languages are, in many cases, easily transported to and run on machines with different processors with little or no modification. For instance the Pascal assignment statement **velocity := accler * time;** can be written on an Apple IIe or a DEC VAX 11/750. However, the internal mechanism that calculates and assigns the result to a place in computer memory will be very different for each computer; the programmer does not have to change his concept of how Pascal works. Instead he just needs to understand that **velocity**, **accler** and **time** represent computer memory locations. Pascal takes care of the details. To the programmer the Apple and VAX "look" like the same computer.

Electronic spreadsheets go a step further by eliminating the computer from the model. High level computer languages, while modeling an idealized computer, still require the programmer to be aware of the machine. The user still has to be concerned with memory locations, variable types, mixed mode calculations, input/output and the like. A spreadsheet provides the user with a general problem solving tool just as Pascal is a problem solving tool for the computer programmer, but the ideal machine is different. Instead of looking like a programming language in the usual sense, an image of an accountant's spreadsheet is presented to the user on the computer's display. Each entry to the sheet is typed into a "cell". The user need only be concerned with how the cells work. As the user enters data and formulae into the cells of the spreadsheet, he is providing instructions for the computer to execute.

The spreadsheet as a programming language has been played down for sound marketing reasons. The last thing software vendors want their perspective customer to think is that he has to be a programmer to use a spreadsheet.

Programming Tool and a Tool for Programmers

Let's compare these two programming languages by solving a simple problem with a Pascal program and do the same task with Multiplan. Consider the Fibbonacci series. The first few terms look like this:

1 1 2 3 5 8 13 21 . . .

The n^{th} term is the sum of the $(n-1)^{th}$ term $(n-2)^{nd}$ term. A program to calculate n terms of the series must do the following:

1. Ask the user how many terms he wishes to calculate.

2. Initialize the first two terms of the series.

3. Set up a mechanism for saving the two previously calculated terms. (i.e., (n-1) and (n-2)).

4. Choose a looping strategy to repeatedly calculate terms until the user specified end point is reached.

5. Print out the series. (A table would be nice.)

Here is a minimal Pascal program that does the job.

```
program fibbo;
 var
  i, limit, newterm : integer;
  last, next2last : integer;
begin
 write('How many terms => ');
 readln(limit);
 last := 1;
 next2last := 1;
 for i := 1 to limit do
  begin
   if i < 3 then
    newterm := 1
   else
    begin
     newterm := last + next2last;
     next2last := last;
     last := newterm;
    end;
   writeln('Term # ', i : 4, newterm : 7);
  end;
end.
```

Listing 1.

Here is the same calculation done with Multiplan.

1. Choose a column.

2. Enter 1 in the first 2 rows as starting values.

3. Enter the series formula in relative cell notation. **(= R[-1]C + R[-2]C).** With the Macintosh implementation of Multiplan, these cell locations are calculated by pointing to the cells with the cursor.

4. **FILL DOWN** the formula in row three for as many terms as desired. Some spreadsheets refer to this as "copying down."

5. You have a nice table of values!!!

Listing 2 is the spreadsheet described above presented in a more organized way.

Title: **fibbo** Settings: **Default**

Purpose: **To illustrate the use of Multiplan vs Pascal for a simple programming task.**

What it does: **Calculates n terms of the Fibbonacci series. The variable n is determined by how many cells from cell R3C2 are filled down column 2.**

Cell	Name	Contents	Format	Remarks
R1C2	--	1		
R2C2	--	1		
R3C2	--	-R[-1]C + R[-2]C		
R4C2:R#C2		*** FILL DOWN FROM R3C2 ***		FOR nTERMS OF THE SERIES LET i = n - 3

Listing 2.

You can use this chart to enter this Multiplan template. This format will be used for the example sheets presented thoughout this article.

Explanation of columns:

Cell - The physical location of the cell on the spreadsheet. These are the **actual** coordinates of the cell. R1C2 is Row 1 Column 2.

Name - Mutiplan allows the user to assign a mnemonic reference to a cell location or group of cell locations. This cell reference, name, can have as many as 31 characters. The first character must be a letter, all subsequent characters can be any combination of numerals (0..9), letters (A..Z,a..z), period (.) or underscore (_).

Contents - As the name implies this is what the cell contains, as opposed to what it might display. The contents can be a number, text (for labels and titles) or a formula.

Format - The format determines how the cells present state is displayed. For example, textual material can be centered in the cell or right justified. Numerical values can be displayed as dollars with a preceding dollar sign ($) and two decimal places. Values can be expressed in scientific notation or as a bar graph.

Remarks - Any notes that may clarify what is going on in this cell.

Advantages of Using a Spreadsheet

Why is it so much easier to calculate the Fibbonacci series with the spreadsheet?

• Keyboard input and screen formatting routines are taken care of for you. In the Pascal program (Listing 1.) a significant fraction of the program instructions are devoted to input/output jobs. (This is a minimal Pascal implementation. Typically a much greater proportion of a program is devoted to input/output.) If you want to do a quick calculation or try out a new or questionable algorithm, Multiplan doesn't burden you with user interface overhead. Of course, you're stuck with the fixed format of the spreadsheet. The trade off in many instances is well worth it.

• The spreadsheet is completely interactive. It provides line by line error checking. If you enter a formula which is syntactically incorrect you don't have to compile the whole program to find out. Every change that's made is reflected by an immediate change on the sheet. This encourages exploring many different ideas. You can get answers quickly.

• The cell format encourages partitioning your task into small modules. A cell can only hold so much. The spreadseet layout forces an organization of the modules as well.

• Multiplan• allows cells or groups of cells to be given meaningful names. The expression 0.5* R3C3* R2C3 ^ 2 becomes 0.5 * GRAVITY * TIME ^ 2.

• Multiplan has a comprehensive collection of functions well suited for science and engineering applications including transcendental and string manipulation functions. Advanced spreadsheets such as Multiplan allow branching operations and boolean arithmetic. For example, the **IF()** function permits one to perform either of two calculations depending on the results of a calculation in another cell. (See programming examples for details.)

• Selected cells of the sheet can be protected so that formulae they contain cannot be altered by the user. "Stand alone" applications can be developed complete with user prompts and error messages.

• Built-in table lookup routines and index functions make simple matrix manipulation possible. The **INDEX()** function allows access to a rectangular group of cell via local indices. A formula such as **=INDEX(R5C7:R8C9,1,2)** returns the value contained in cell R5C8.

• Calculations requiring iterative solutions can be done. Iteration is a way of arriving at an answer to a calculation by taking an initial guess at the answer then using an algorithm to come up with better and better guesses. To do iteration, Multiplan allows you to make circular references between cells. (The value of cell A may depend upon the result of the calculation in cell B. Cell B's value may include a reference to cell A. Sound confusing? We'll discuss this process in detail later.)

• Results of calculations of one sheet can be transferred to another sheet. If the results of one sheet's calculation change, the change will automatically be reflected in the other sheet that uses these results.

Limitations

There are, of course, limitations to using the spreadsheet as a programming language. As the sheet fills up recalculation takes longer. You cannot force a recalculation of only part of the sheet, but, it is possible to break up the sheet into smaller pieces (i.e., different files) then paste a link the results of one sheet to the next.

The fixed format is also a limiting factor. The position of cells and choice of a limited selection of cell formats may prevent certain applications from being implemented.

The implementation of iteration on Multiplan forces the order of calculation of cells to occur in a way which restricts the positioning of formulae on the sheet. Sometimes the planning required to get the formulae in the correct order takes longer than using an alternate medium for finding a solution.

An Annotated Collection of Examples

Let's look at some examples of programs done with Multiplan. Feel free to use them as starting points from which to expand with your own ideas.

These examples were all programmed on the Apple Macintosh version of Multiplan. Multiplan is also available for the Apple II PC-DOS, MS-DOS and CP/M. All of these sheets can be transferred, with minor modifications, to any implementation of Multiplan. Some of these examples can also be adapted to other spreadsheet programs, such as Visicalc, Lotus 1-2-3, and SuperCalc.

Please refer to Appendix A for a complete listing of the samples described below.

1. Naming cells

   a. **freefall(1).** Multiplan allows the user to refer to a cell location in a variety of ways. Formulae make more sense when the quantities being manipulated have names that mean something to the user. That is, **velocity - dist/time** means more than to the user than **-R1C3/R5C6.**

      This example looks at the dynamics of an object falling vertically under the influence of a gravitational acceleration. Cells in the first five rows specify the physical parameters of the phenomenon. The next thirty rows contain identical sets of formula that are copied (Filled Down) from the first four columns of row seven.

2. Plotting graphs

   a. **sinecurve(2).** The **REPT()** function is used here to generate a number of spaces (blanks) proportional to the value of the function calculated in the adjacent cell. An asterisk(*) is concatenated to the spaces using the "&" operator.

For the sine function two plotting constants are used. Only an integral number of spaces can be plotted. You can't plot 0.78 spaces. Since the absolute value of the sine function is always less than or equal to one, the value that is plotted must be scaled up (multiplied) by some constant. In this case 20 was chosen. (It seemed like a nice number). That means when the value of the sine is one, twenty spaces and an asterisk will be plotted.

The user must also account for the instances when the sine function is less than zero. A plot offset must be added to the scaled value. The offset constant of 22 was chosen. This implies that when the value of the function is "-1", two spaces and an asterisk will be plotted.

b. **barGraph(3).** This is one of my favorite programs. It is based upon concepts developed by Alan Kay in his September 1984 *Scientific American*[1] article. The program is a completely self-scaling histogram. As the data in row 12 are changed the plot will rescale and change. Watching the graph change is half the fun. The amazing thing about this template is that only three basic formulae are used.

3. Setting up a stand alone program for another user.

a. **numguess(4).** The spreadsheet can also be used to create a program that is intended to be used by someone other than the developer. Areas of the sheet can be protected so that the users cannot tamper with the formulae. This template also demonstrates how to provide prompts for the user, that is create a sort of user interface. Thus **numguess(4)** demo provides a point of departure for one considering developing a sheet to be used by someone else.

---

1. Kay, Alan, "Computer Software", *Scientific American* 251,3 (Sept 1984)

This program is general. If the value of LIMIT or NUMBER is changed, the hint table "automatically" readjusts so that the correct message appears at the HINT==> location. In a real application the hints would be hidden from view, of course.

4. Iterative calculations.

The next three examples use the spreadsheet in a different way from the four sheets illustrated above. Typically, the cell values in those sheets remain the same until the user alters the value of a cell on which other cell values depend. For a short period of time during recalculation there is some observable activity until the cells reach a new equilibrium. In the following sheets there is continuous activity as calculations procede. The spread sheet "looks" like it's "running".

a. **newton(5).** This sheet, **newton(5)**, and the next one, **fibbo.ser(6)**, obtain a solution by the process of iteration. Newton's method for finding the square root is an archetypical example of an iterative solution. First enter a number, called the argument. An initial guess for the squareroot of the argument is made. This initial guess, which we'll refer to as the old guess, is used to calculate a new guess via the simple relation. newguess = 0.5 *((argument/oldguess) + oldguess).

The newguess can then be used as the oldguess to get a better guess. We can stop this process in several ways. One way is to use the definition of the square root. Multiply newguess by itself and see if the product is close enough to the argument to be an acceptable solution. One can compare the old guess to the new guess and continue to iterate until the absolute value of the difference between guesses is less than a pre-determined value, say 0.001. If nothing is specified explicitly by the user Multiplan will stop iterating when the values in the cells differ by less than 0.001 between successive iterations.

Another issue that needs to be addressed is how to start the iteration process. An inital value for a guess must be inserted

into the iteration process. After the start, the new calculated guess must be obtained from a different location. Keep in mind that if the user enters a number into a cell it obliterates any existing formula that was there. So an initial guess has to be "plucked" out of a cell not used to calculate subsequent guesses. One way of implementing this is illustrated in **newton(5)** see cells R9C3 to R12C3.

Multiplan keeps a record of how many iterations it has performed. The current count is accessable through the **ITERCNT()** function. **ITERCNT()** returns the current iteration number. The first time around is defined as one (1). The count number is used to direct the selection of values used in the calculation. Examine R11C3:

=IF(ITERCNT()=1, R[-1],0.5*((ARG/R[+1]C)+R[+1]C))

In this case the value in R10C3 is used for the first iteration only. All other iterations use the value in R12C3.

b. **fibbo.ser(6)** This is another way of programming the fibbonacci series. It looks very much like our Pascal programming example. Note that the results of each iteration is larger that the previous one. Multiplan will not stop this automatically! The cell R12C2 contains a user entered constant that is used by the cell below it in boolean calculation. The value of this expression will be TRUE or FALSE.

=ITERCNT()=R[-1]

Use the set completion test option to mark R13C2 to be used as the stop criterion. When the result of the calculation in that cell is true, the program will stop.

5. The animation (yes, animation!. . . .kind of . . .).

a. **moving mess(7)** Watch the letters march across the screen.

A Spreadsheet Is a Different Kind of Programming Language

A spreadsheet seems different from many other programming languages because it is, in a sense, object oriented. A spreadsheet is a collection of things called cells. If the user assigns a number or text to the cell, the cell will display that number or text no matter what the states of any other cells on the sheet assume. The state of a cell can be forced to change if its state is dependent on the state of another cell or group of cells. Consider a cell that will display the sum of the numbers displayed in the column of cells above it. The state of this cell is dependent upon the states of other cells on the sheet. (See figure 1.)

The spreadsheet is non-procedural. Each cell assumes a specific state depending upon the properties that the user assigns to it. Most of the time the sheet is in a kind of equilibrium; that is, the sheet is static. Each cell has assumed a specific state dictated by its assigned properties. Unlike a procedural language such as BASIC or Pascal, the spreadsheet is static most of the time. A BASIC program is continually changing its state as each statement in the program is sequentialy executed. There is only a relatively short period of time when the spreadsheet is in transition from one state to another. This transition can only occur when the user modifies the properties of a cell. (i.e., changes a formula or changes a number or string that other cells depend on).

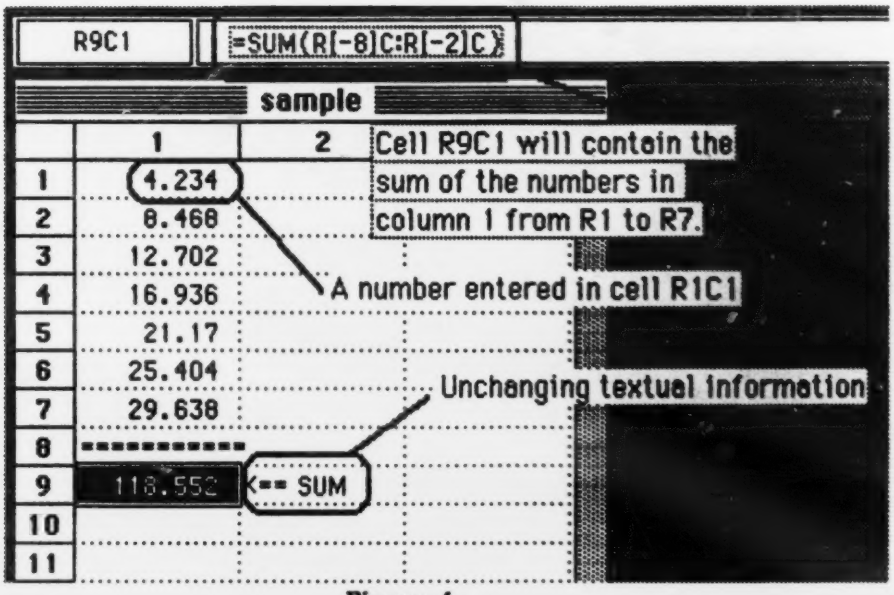

R9C1
=SUM(R[-8]C:R[-2]C)
sample
1
2
Cell R9C1 will contain the
sum of the numbers in
column 1 from R1 to R7.
1
4.234
2
8.468
3
12.702
4
16.936
5
21.17
6
25.404
7
29.638
8
9
118.552
<== SUM
10
11
A number entered in cell R1C1
Unchanging textuel information

Figure 1.

APPENDIX A

EXAMPLE TEMPLATES

File Edit Select Format Options Calculate

R8C2 | =SQRT(2*(INIT_HEIGHT-RC[-1])*GRAV)

freefall(1)

	1	2	3	4	5	6
1		FALLING BODY				
2					© Stan Cohen 19 Nov 84	
3	MASS	0.5	INIT HEIGHT	50		
4	ACC OF GRAV	9.8				
5	STEP SIZE	2				
6	X	VEL	PE	KE	TIME	
7	50	0	245	0	0	
8	48	6.2609903	235.2	9.8	0.6388766	
9	46	8.8543774	225.4	19.6	0.9035079	
10	44	10.844353	215.6	29.4	1.1065667	
11	42	12.521981	205.8	39.2	1.2777531	
12	40	14	196	49	1.4285714	
13	38	15.336232	186.2	58.8	1.5649216	
14	36	16.565023	176.4	68.6	1.6903085	
15	34	17.708755	166.6	78.4	1.8070158	

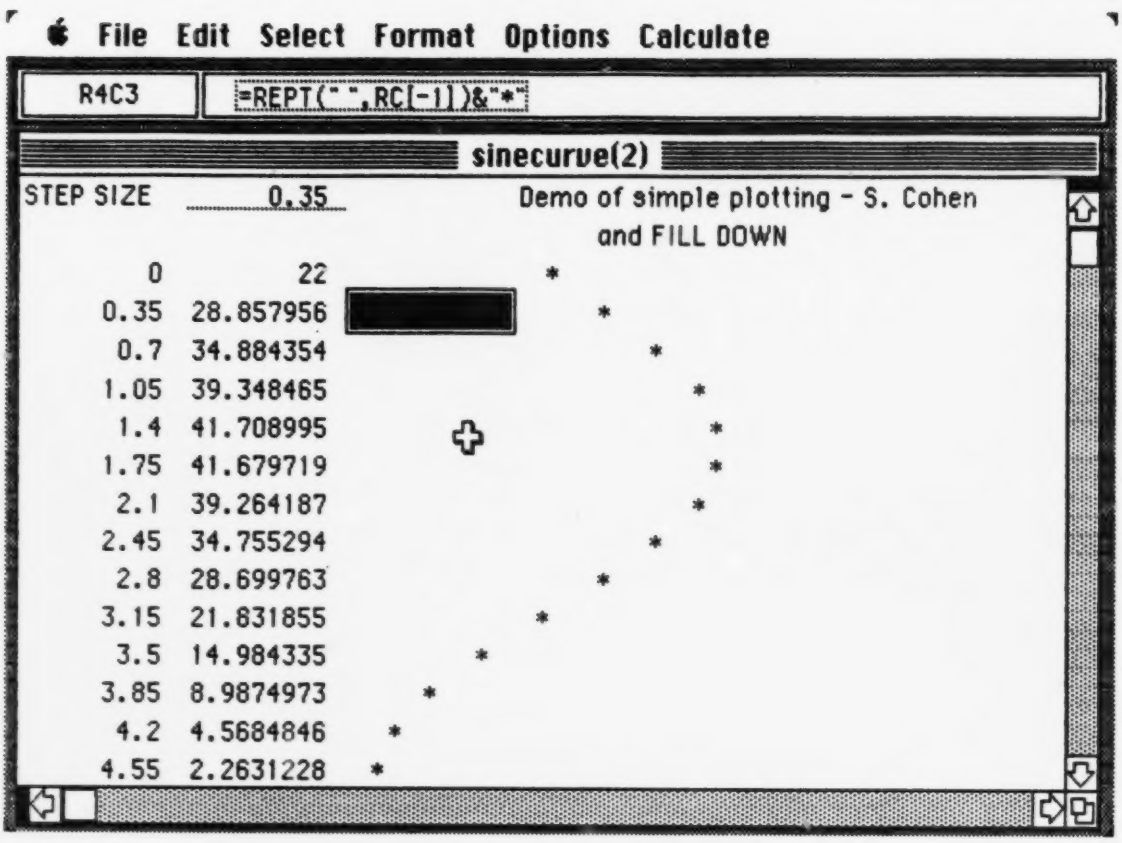
Multiplan Template Set-Up sheet

Title: **freefall(1)** Settings: **Default**

Purpose: **To illustrate the use of mnemonic labeling of variables in Multiplan.**

What it does : **Tracks the position, kinetic energy and potential energy of a freely falling body of mass "m" dropped from a height specified by the user. Generates a table of vertical height, velocity, elapsed time from release, and kinetic and potential energy of the mass at that height.**

Cell	Name	Contents	Format	Remarks
R3C1	--	MASS		
R4C1	--	ACC OF GRAV		
R5C1	--	STEP SIZE		
R6C1	--	X		
R7C1	--	=(INIT_HEIGHT -(ROW()-7) * STEP_SIZE		DISTANCE FROM RELEASE HEIGHT
R8C1:R38C1		**FILLED DOWN FROM R7C1**		
R3C2	MASS	0.5		USER INPUT
R4C2	GRAV	9.8		CONSTANT
R5C2	STEP_SIZE	2		USER INPUT
R6C2	--	VEL		
R7C2	--	=SQRT(2*INIT_HEIGHT-RC[-1])*GRAV		VELOCITY AT RC[-1]
R8C2:R38C2		**FILLED DOWN FROM R7C2**		
R1C3	--	FALLING BODY		
R3C3	--	INIT HEIGHT		
R6C3	--	PE		
R7C3	--	=MASS* GRAV*RC[-2]		POTENTIAL ENERGY AT RC[-2]
R8C3:R38C3		**FILLED DOWN FROM R7C3**		
R3C4	INIT_HEIGHT	50		USER INPUT
R6C4	--	KE		
R7C4	--	=0.5*MASS*RC[-2]*RC[-2]		KINETIC ENERGY AT RC[-3]
R8C4:R38C4		**FILLED DOWN FROM R7C4**		
R6C5	--	TIME		
R7C5	--	=SQRT(2*(INIT_HEIGHT - RC[-4])/GRAV)		ELAPSED TIME SINCE RELEASE
R8C5:R38C5		**FILLED DOWN FROM R7C5**		


File Edit Select Format Options Calculate
R4C3
=REPT(" ",RC[-1])&"*"
sinecurve(2)
STEP SIZE 0.35
Demo of simple plotting - S. Cohen
and FILL DOWN
0 22
0.35 28.857956
0.7 34.884354
1.05 39.348465
1.4 41.708995
1.75 41.679719
2.1 39.264187
2.45 34.755294
2.8 28.699763
3.15 21.831855
3.5 14.984335
3.85 8.9874973
4.2 4.5684846
4.55 2.2631228

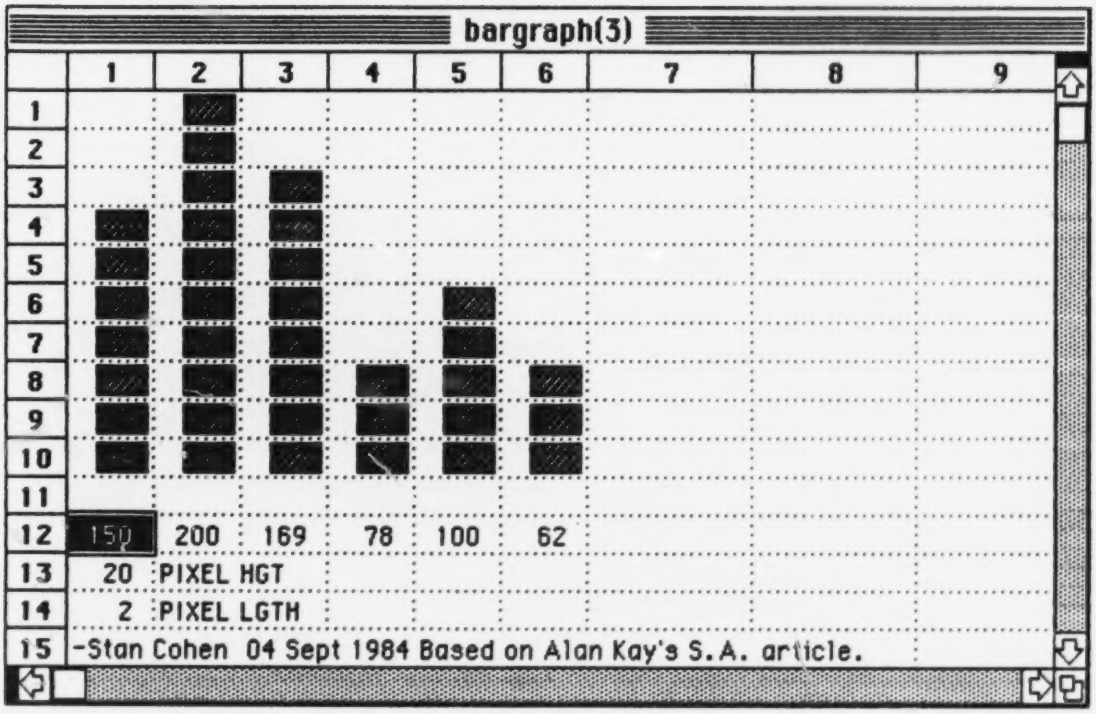
Multiplan Template Set-Up sheet

Title: **sinecurve(2)** Settings: **Default**

Purpose: **To illustrate the use of Multiplan to plot functions.**

What it does : **Calculates the values for the sine function. An even abcissa increment is assumed. Each point is plotted in a successive row. Any function can be plotted by changing the formula in R3C2 and filling down. For any serious work the scale factor and offset values should be calculated instead of explictly placing them in the cell formulae.**

Cell	Name	Contents	Format	Remarks
R1C1	--	STEP SIZE		
R3C1	--	=(ROW()-3)*STEP		
R4C1:R17C1	*** FILL DOWN FROM R3C1 ***			
R1C2	STEP	0.5		USER INPUT
R3C2		=SIN(RC[-1])*20+22		20 IS THE SCALE FACTOR 22 IS THE OFFSET
R4C2:R17C2	*** FILL DOWN FROM R3C2 ***			
R3C3		=REPT(" ",RC[-1])&"*"		
R4C3:R17C3	*** FILL DOWN FROM R3C3 ***			


bargraph(3)
1 2 3 4 5 6 7 8 9
1
2
3
4
5
6
7
8
9
10
11
12 150 200 169 78 100 62
13 20 PIXEL HGT
14 2 PIXEL LGTH
15 -Stan Cohen 04 Sept 1984 Based on Alan Kay's S.A. article.

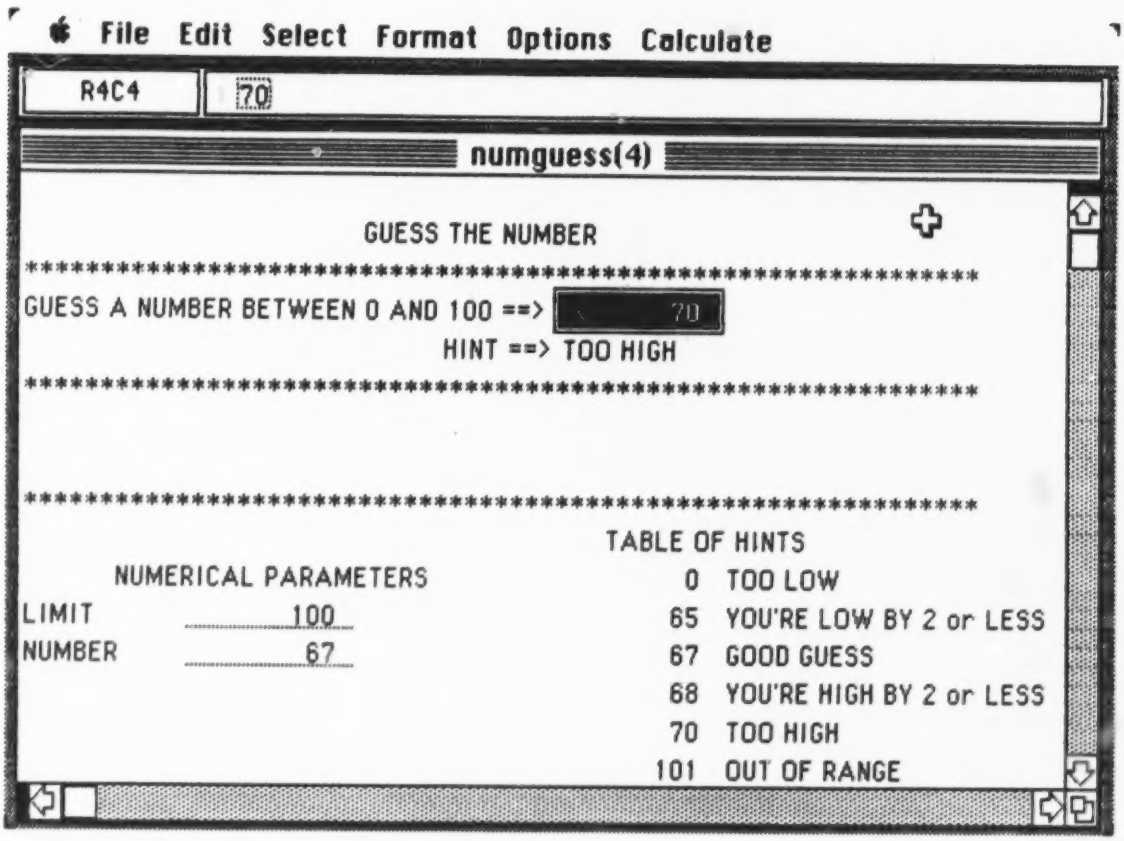
Multiplan Template Set-Up sheet

Title: **barGraph(3)** Settings: **Default**

Purpose: **Generate a histogram for a given data set.**

What is does : **Uses the algorithm proposed by Alan Kay in his September 1984 Scientific American article. The graph is simple to set up and self scaling.**

Cell	Name	Contents	Format	Remarks
R1C1	--	=IF((11-ROW())*PIXHGT <= R12C1,PIXLEN,0)	BARGRAPH	
R2C1:R10C1	--	*** FILL DOWN FROM R1C1 ***		
R12C1	DATA	150		USER ENTRY
R13C1	PIXHGT	=MAX(DATA)/10		
R14C1	PIXLEN	2		A NEGATIVE NUMBER MAKES LIGHT BARS A POSITIVE ONE MAKES DRK BARS
R1C2	--	=IF((11-ROW())*PIXHGT <= R12C2,PIXLEN,0)	BARGRAPH	
R2C2:R10C2	--	*** FILL DOWN FROM R1C2 ***		
R12C2	DATA	200		USER ENTRY
R1C3	--	=IF((11-ROW())*PIXHGT <= R12C3,PIXLEN,0)	BARGRAPH	
R2C3:R10C3	--	*** FILL DOWN FROM R1C1 ***		
R12C3	DATA	169		USER ENTRY
R1C4	--	=IF((11-ROW())*PIXHGT <= R12C4,PIXLEN,0)	BARGRAPH	
R2C4:R10C4	--	*** FILL DOWN FROM R1C4 ***		
R12C4	DATA	250		USER ENTRY
R1C5	--	=IF((11-ROW())*PIXHGT <= R12C5,PIXLEN,0)	BARGRAPH	
R2C5:R10C5	--	*** FILL DOWN FROM R1C1 ***		
R12C5	DATA	100	USER ENTRY	
R1C6	--	=IF((11-ROW())*PIXHGT <= R12C6,PIXLEN,0)	BARGRAPH	
R2C6:R10C6	--	*** FILL DOWN FROM R1C6 ***		
R12C1	DATA	80	USER ENTRY	


File Edit Select Format Options Calculate
R4C4
70
numguess(4)
GUESS THE NUMBER
GUESS A NUMBER BETWEEN 0 AND 100 ==> 70
HINT ==> TOO HIGH
NUMERICAL PARAMETERS
LIMIT 100
NUMBER 67
TABLE OF HINTS
0 TOO LOW
65 YOU'RE LOW BY 2 or LESS
67 GOOD GUESS
68 YOU'RE HIGH BY 2 or LESS
70 TOO HIGH
101 OUT OF RANGE

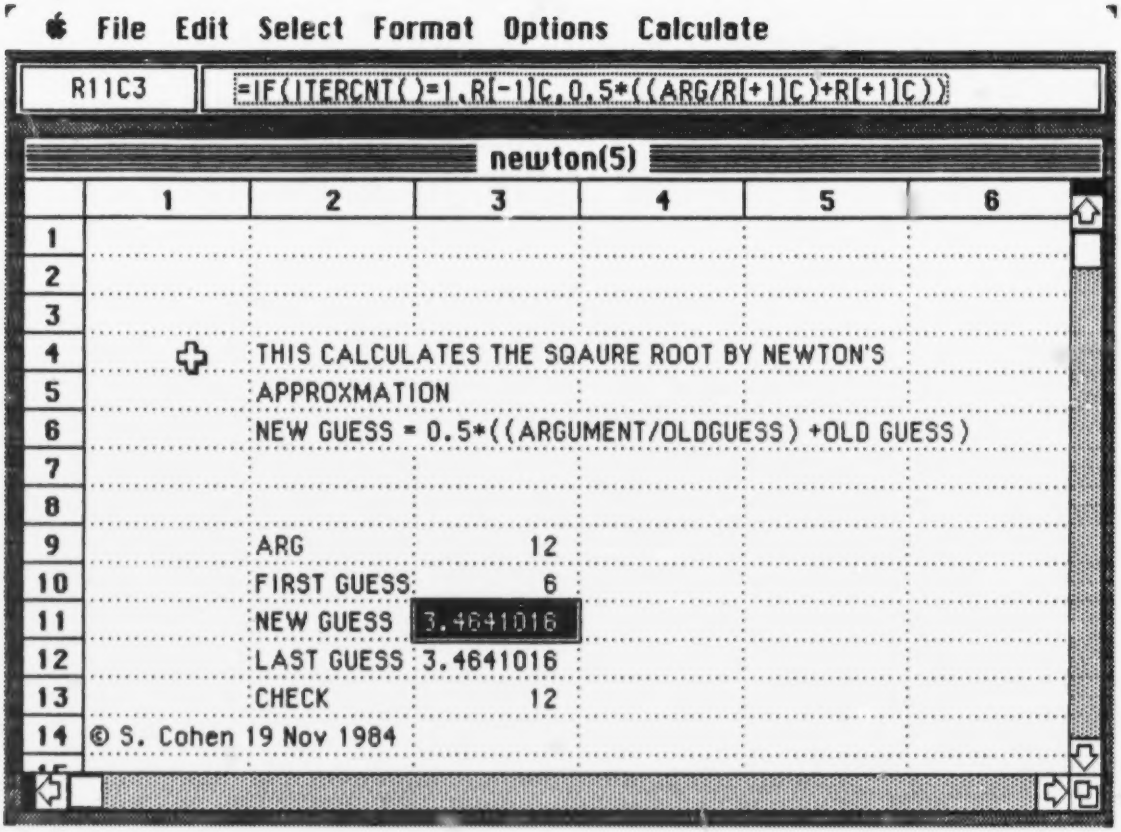
Multiplan Template Set-Up sheet

Title: **numguess(4)** Settings: **Default**

Purpose: **To illustrate the use of Multiplan as a stand alone program with user prompts**

What it does : **This program is a prototype for developing a template for a naive user (who is someone other than the developer).**

Cell	Name	Contents	Format	Remarks
R3C1	--	=REPT("*",63)		DECORATION
R4C1	--	="GUESS A NUMBER BETWEEN 0 AND"&FIXED(LIMIT,0)&"==>"		
R6C1	--	=REPT("*",63)		DECORATION
R12C1	--	LIMIT		
R13C1	--	NUMBER		
R11C2	--	NUMERICAL PARAMETERS		
R12C2	LIMIT	100		PROGRAMMER INPUT
R13C2	NUMBER	55		PROGRAMMER INPUT
R2C3	--	GUESS THE NUMBER		TITLE
R5C3	--	HINT ==>		
R4C4	--	56		USER ENTRY
R5C4	--	=LOOKUP(R[-1]C,HINTS)		
R10C4	--	TABLE OF HINTS		
R11C4	HINTS	=0		
R12C4	HINTS	=NUMBER -2		
R13C4	HINTS	=NUMBER		
R14C4	HINTS	=NUMBER +1		
R15C4	HINTS	=NUMBER +3		
R16C5	HINTS	=LIMIT +1		
R11C5	HINTS	=TOO LOW		
R12C5	HINTS	=YOU'RE LOW BY TWO OR LESS		
R13C5	HINTS	=GOOD GUESS		
R14C5	HINTS	=YOU'RE HIGH BY TWO OR MORE		
R15C5	HINTS	=YOU'RE HIGH		
R16C5	HINTS	=OUT OF RANGE		


File Edit Select Format Options Calculate
R11C3
=IF(ITERCNT()=1,R[-1]C,0.5*((ARG/R[+1]C)+R[+1]C))
newton(5)
1 2 3 4 5 6
1
2
3
4 THIS CALCULATES THE SQAURE ROOT BY NEWTON'S
5 APPROXMATION
6 NEW GUESS = 0.5*((ARGUMENT/OLDGUESS) +OLD GUESS)
7
8
9 ARG 12
10 FIRST GUESS 6
11 NEW GUESS 3.4641016
12 LAST GUESS 3.4641016
13 CHECK 12
14 © S. Cohen 19 Nov 1984

Multiplan Template Set-Up sheet

Title: **newton(5)**

Settings: **Iterate**
Manual calculation

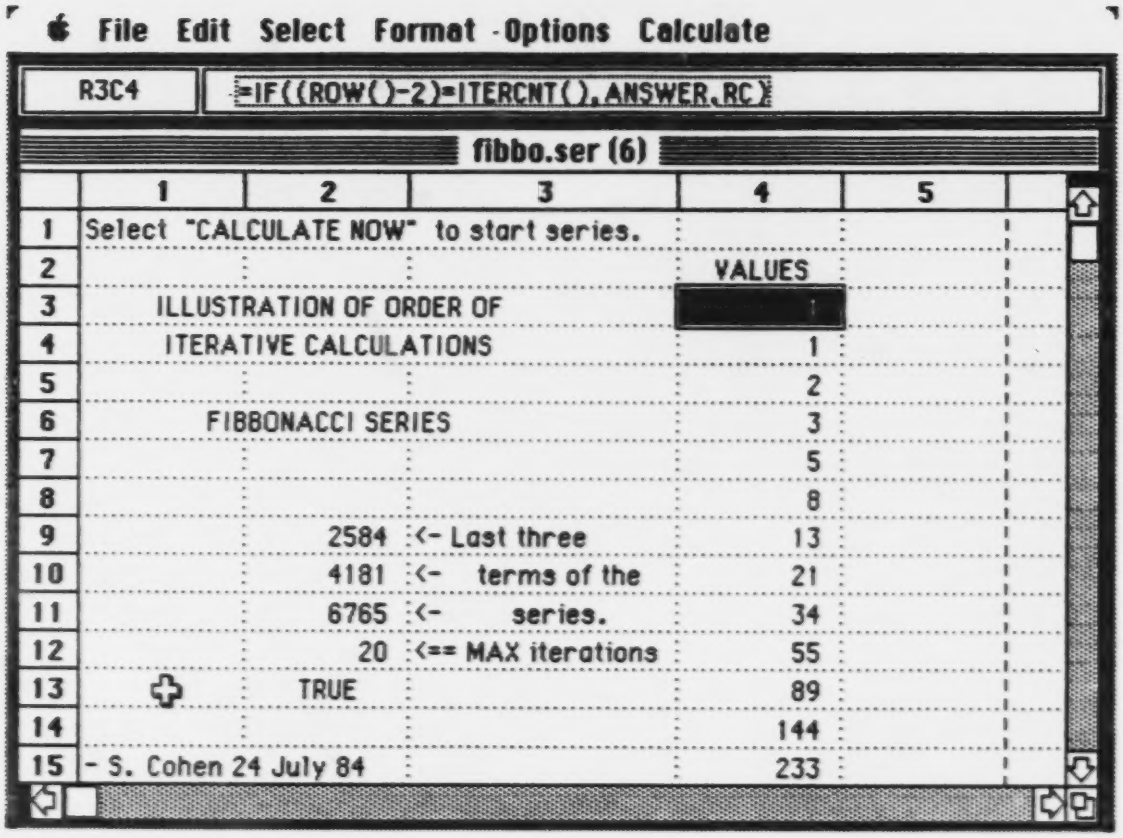
Purpose: **To illustrate the use of iteration to solve a problem with Multiplan.**

What is does : **Calculates the square root of a number (argument) by suggessive approximation using the relationship,**

new_guess = 0.5 * ((argument/old_guess) + old_guess.

The intial old_guess is forced to be argument/2. If new_guess is not close enough to the real value of the square root, new_guess is used as old_guess and another new_guess is calculated. The process is typically stopped when the absolute value of the difference of two successive guesses is less than some small value, say 0.0001. Multiplan will stop automatically when the difference between answers from the last iteration and current iteration is less than 0.001.

Cell	Name	Contents	Format	Remarks
R9C2	--	ARG		
R10C2	--	FIRST GUESS		
R11C2	--	NEW GUESS		
R12C2	--	LAST GUESS		
R13C2	--	CHECK		
R9C3	ARG	100		USER INPUT
R10C3	--	=ARG/2		
R11C3	--	=IF(ITERCNT()=1, R[-1]C, 0.5*((ARG/R[+1])+R[+1]C)		
R12C3	--	R[-1]C		NOTE: R11C3 REFERS TO R12C3 AND R12C3 REFERS TO R11C3. THE OPTION ITERATE MUST BE SELECTED. IF NOT THE ERROR- CAN'T RESOLVE CIRCULAR REFERENCES -WILL BE DISPLAYED.
R13C3	--	=R[-2]C* R[-2]C		


File Edit Select Format Options Calculate
R3C4
=IF((ROW()-2)=ITERCNT(),ANSWER,RC)
fibbo.ser (6)
Select "CALCULATE NOW" to start series.
ILLUSTRATION OF ORDER OF
ITERATIVE CALCULATIONS
FIBBONACCI SERIES
2584 <- Last three
4181 <- terms of the
6765 <- series.
20 <== MAX iterations
TRUE
- S. Cohen 24 July 84
VALUES
1
1
2
3
5
8
13
21
34
55
89
144
233

Multiplan Template Set-Up sheet

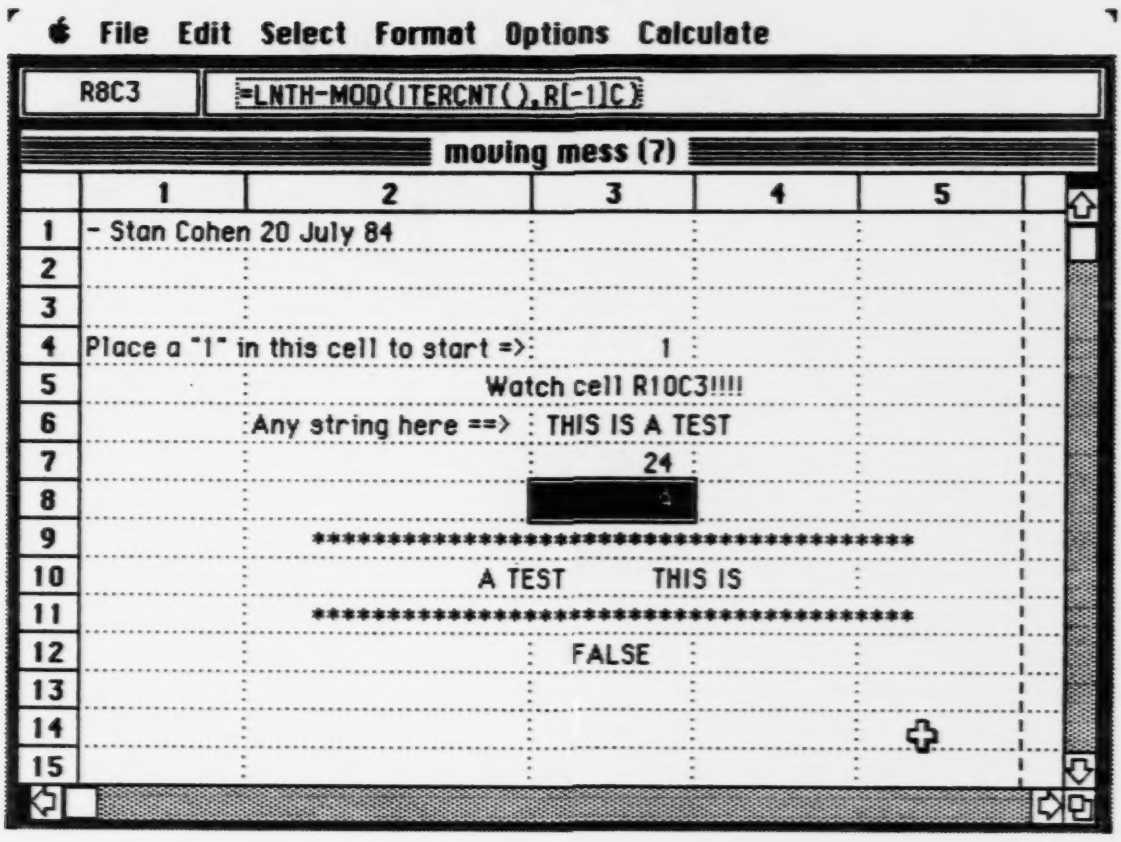
Title: **fibbo.ser(6)**

Settings: **Iterate**
Manual calculation

Purpose: **Generates the fibbonacci series using iteration.**

What it does : **Uses the iteration to calculate the terms of the fibbonacci series in a way similar to using a procedural language.**

Cell	Name	Contents	Format	Remarks
R6C2	--	FIBBONACCI SERIES		
R9C2	--	=R[+1]C		
R10C2	--	=R[+1]C		
R11C2	ANSWER	=IF(ITERCNT()<3, 1,R[-2]C+R[-1]C)		
R12C2	--	20		USER INPUT
R13C2	--	=ITERCNT()=R[-1]		SET COMPLETION TEST
R3C4	--	=IF(ROW()-2=ITERCNT(),ANSWER,RC)		
R4C4:R24C4	--	*** FILL DOWN FROM R3C4 **		


File Edit Select Format Options Calculate
R8C3
=LNTH-MOD(ITERCNT(),R[-1]C)
moving mess (7)
1 2 3 4 5
- Stan Cohen 20 July 84
Place a "1" in this cell to start =>
1
Watch cell R10C3!!!!
Any string here ==>
THIS IS A TEST
24
A TEST THIS IS
FALSE

Multiplan Template Set-Up sheet

Title: **movingmess(7)**

Settings: **Iterate**
Column 2 -- width = 18

Purpose: **To illustrate the use of Multiplan's built-in string functions.**

What it does : **Uses the iteration counter of Multiplan along with built-in string functions to create a moving banner of letters on the spreadsheet. (Actual *practical* uses for this program has yet to be determined.)**

Cell	Name	Contents	Format	Remarks
R4C1	--	Place a '1' in this cell to start =>		
R6C2	--	Any string here =>		
R4C3	--	0		USER INPUT 0 = reset 1 = start
R5C3	--	Watch cell at R10C3	CENTERED	
R6C3	BANNER	THIS IS A TEST		USER INPUT
R7C3	LNTH	=LEN(BANNER)		
R8C3	PNTR	=LNTH-MOD(ITERCNT,R[-1]C)		
R9C3	--	=REPT("*",40)	CENTERED	
R10C3	--	=MID(BANNER,PNTR+1, LNTH-PNTR)& MID(BANNER,1,PNTR)		
R11C3	--	=REPT("*",40)	CENTERED	
R12C3	--	=R[-8]C=0		COMPLETION TEST Note: This is forced to be TRUE or FALSE. If the value in R4C3 is non-zero the program will run "forever".

Multiplan Template Set-Up sheet

Title: **barGraph(3)** Settings: **Default**

Purpose: **Generate a histogram for a given data set.**

What is does : **Uses the algorithm proposed by Alan Key in his September 1984 Scientific American article. The graph is simple to set up and self scaling.**

Cell	Name	Contents	Format	Remarks
R1C1	--	=IF((11-ROW())<=PIXHGT ⇦R12C1,PIXLEN,0)	BARGRAPH	
R2C1:R10C1	--	*** FILL DOWN FROM R1C1 ***		
R12C1	DATA	150		USER ENTRY
R13C1	PIXHGT	=MAX(DATA)/10		
R14C1	PIXLEN	2		A NEGATIVE NUMBER MAKES LIGHT BARS A POSITIVE ONE MAKES DRK BARS
R1C2	--	=IF((11-ROW())<=PIXHGT ⇦R12C2,PIXLEN,0)	BARGRAPH	
R2C2:R10C2	--	*** FILL DOWN FROM R1C2 ***		
R12C2	DATA	200		USER ENTRY
R1C3	--	=IF((11-ROW())<=PIXHGT ⇦R12C3,PIXLEN,0)	BARGRAPH	
R2C3:R10C3	--	*** FILL DOWN FROM R1C1 ***		
R12C3	DATA	169		USER ENTRY
R1C4	--	=IF((11-ROW())<=PIXHGT ⇦R12C4,PIXLEN,0)	BARGRAPH	
R2C4:R10C4	--	*** FILL DOWN FROM R1C4 ***		
R12C4	DATA	250		USER ENTRY
R1C5	--	=IF((11-ROW())<=PIXHGT ⇦R12C5,PIXLEN,0)	BARGRAPH	
R2C5:R10C5	--	*** FILL DOWN FROM R1C1 ***		
R12C5	DATA	100	USER ENTRY	
R1C6	--	=IF((11-ROW())<=PIXHGT ⇦R12C6,PIXLEN,0)	BARGRAPH	
R2C6:R10C6	--	*** FILL DOWN FROM R1C6 ***		
R12C1	DATA	80	USER ENTRY	

Multiplan Template Set-Up sheet

Title: **barGraph(3)** Settings: **Default**

Purpose: **Generate a histogram for a given data set.**

What is does : **Uses the algorithm proposed by Alan Kay in his September 1984 Scientific American article. The graph is simple to set up and self scaling.**

Cell	Name	Contents	Format	Remarks
R1C1	--	=IF((11-ROW())*PIXHGT <= R12C1,PIXLEN,0)	BARGRAPH	
R2C1:R10C1	--	*** FILL DOWN FROM R1C1 ***		
R12C1	DATA	150		USER ENTRY
R13C1	PIXHGT	=MAX(DATA)/10		
R14C1	PIXLEN	2		A NEGATIVE NUMBER MAKES LIGHT BARS A POSITIVE ONE MAKES DRK BARS
R1C2	--	=IF((11-ROW())*PIXHGT <= R12C2,PIXLEN,0)	BARGRAPH	
R2C2:R10C2	--	*** FILL DOWN FROM R1C2 ***		
R12C2	DATA	200		USER ENTRY
R1C3	--	=IF((11-ROW())*PIXHGT <= R12C3,PIXLEN,0)	BARGRAPH	
R2C3:R10C3	--	*** FILL DOWN FROM R1C1 ***		
R12C3	DATA	169		USER ENTRY
R1C4	--	=IF((11-ROW())*PIXHGT <= R12C4,PIXLEN,0)	BARGRAPH	
R2C4:R10C4	--	*** FILL DOWN FROM R1C4 ***		
R12C4	DATA	250		USER ENTRY
R1C5	--	=IF((11-ROW())*PIXHGT <= R12C5,PIXLEN,0)	BARGRAPH	
R2C5:R10C5	--	*** FILL DOWN FROM R1C1 ***		
R12C5	DATA	100	USER ENTRY	
R1C6	--	=IF((11-ROW())*PIXHGT <= R12C6,PIXLEN,0)	BARGRAPH	
R2C6:R10C6	--	*** FILL DOWN FROM R1C6 ***		
R12C1	DATA	80	USER ENTRY	